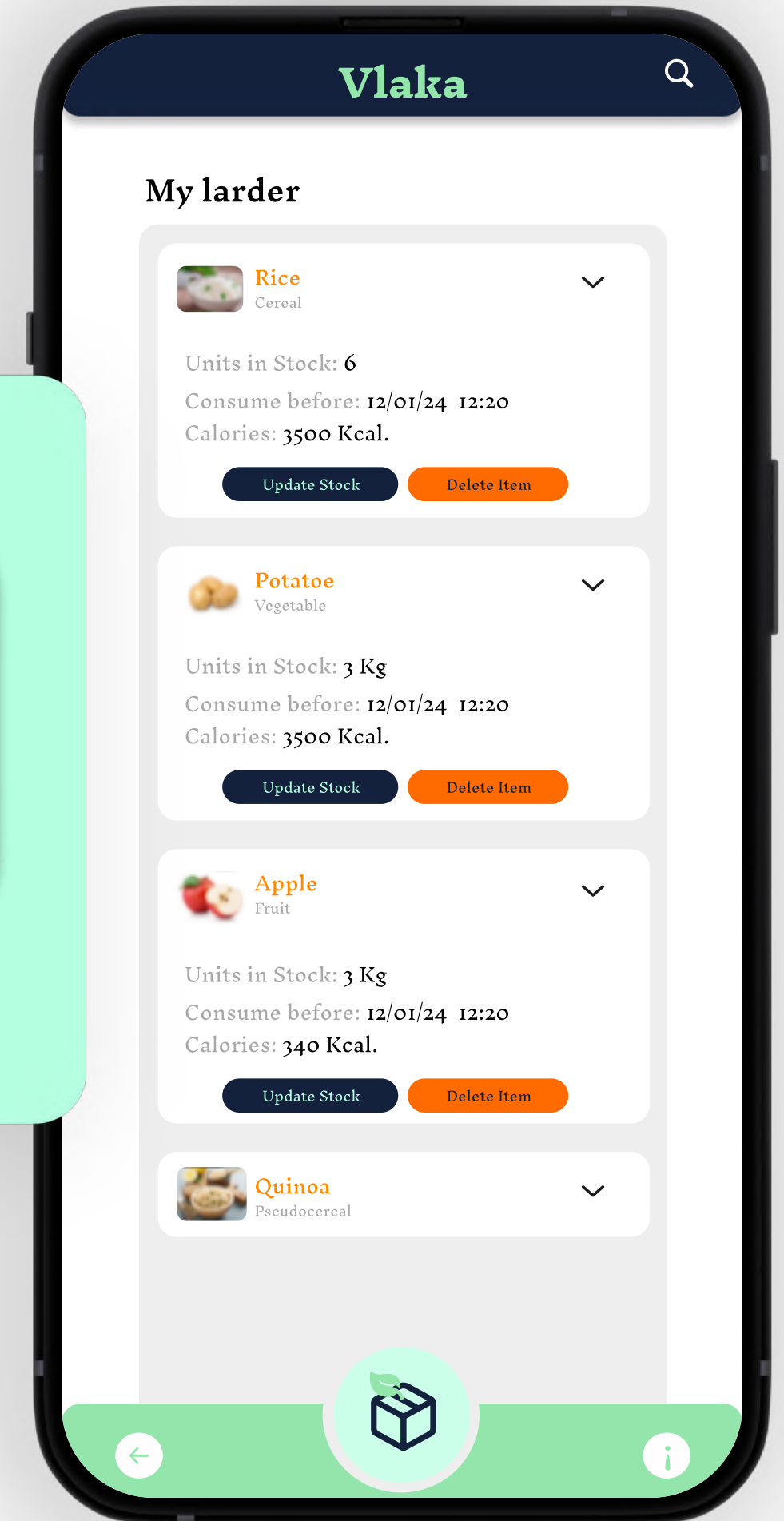
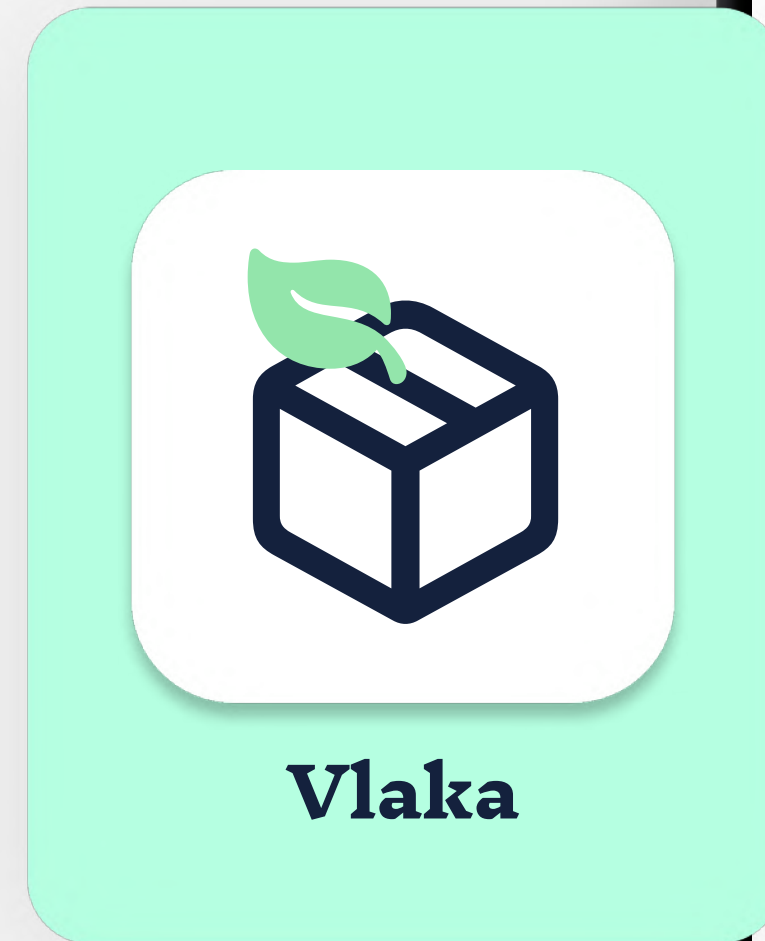

Vlaka (Virtual Larder Assistant Kotlin App)

Despensa Virtual




Índice



Integrantes

01



Problema

02




Solución

03



Preview Mockups

04



Código

05

Mejoras

06

Financiación

07



Integrantes



Edwin March A.

Soy una persona a la que le gusta aprender cosas nuevas y he encontrado un camino al realizar este curso. Me ha apasionado el mundo del desarrollo, pero nunca tuve el valor de intentarlo porque pensaba que todo era demasiado difícil para mí. Pero después de completar este curso me he dado cuenta de que puedo aprender, ¡ahora todo depende de mí!



Adrian Fernández Herrero

Cuando me enseñaron C cambio mi perspectiva así que me centre en aprender otros lenguajes ya que desde ese momento sentí la curiosidad y la necesidad de aprender como funcionaban los lenguajes tanto en back como en front además con el tiempo me he dado cuenta que me apasiona y disfruto mucho desarrollando Apps

Problema



¿Cansado de no saber si tienes ese producto en la despensa o de si no esta caducado?

Problema

¿Quieres información en vivo sobre lo que hay en tu despensa ?
¿No recuerdas si tienes ese alimento en tu despensa?
¿Quieres una mejor gestión de tus alimentos?
¿No recuerdas la caducidad de un producto?
¿Quieres saber el stock de ellos?



Vlaka

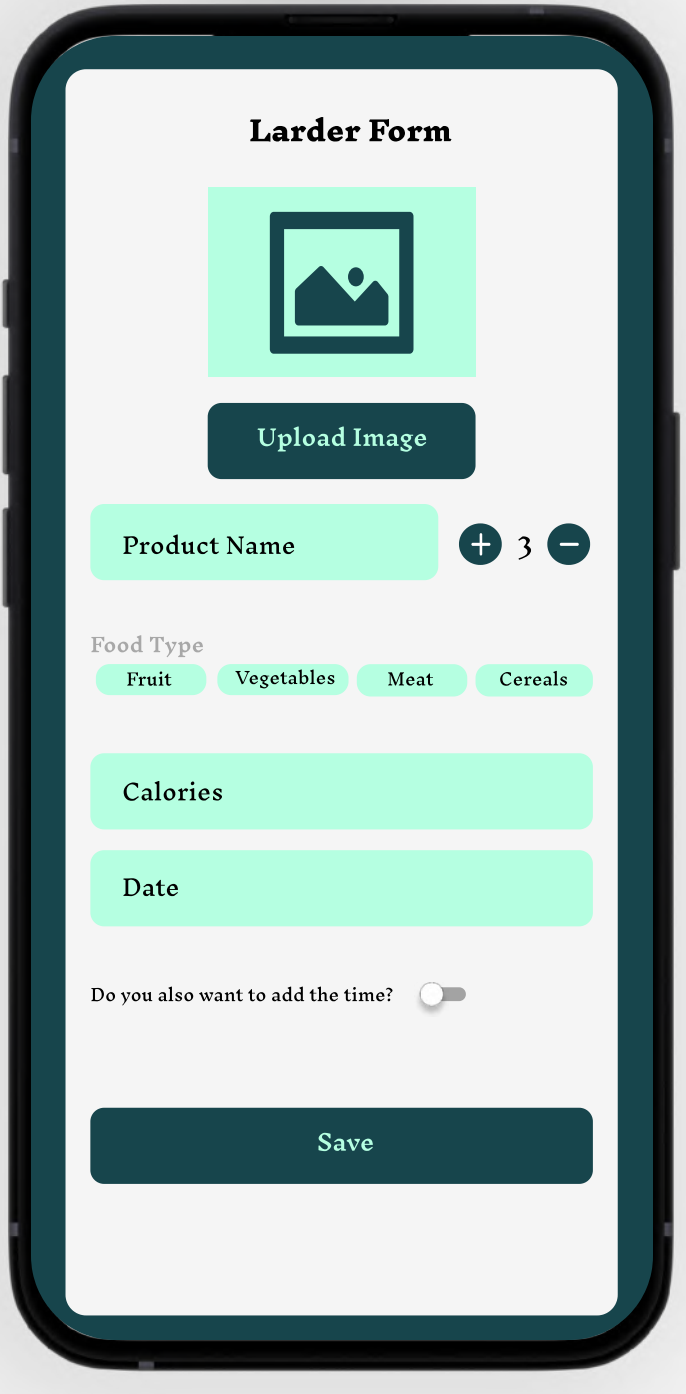
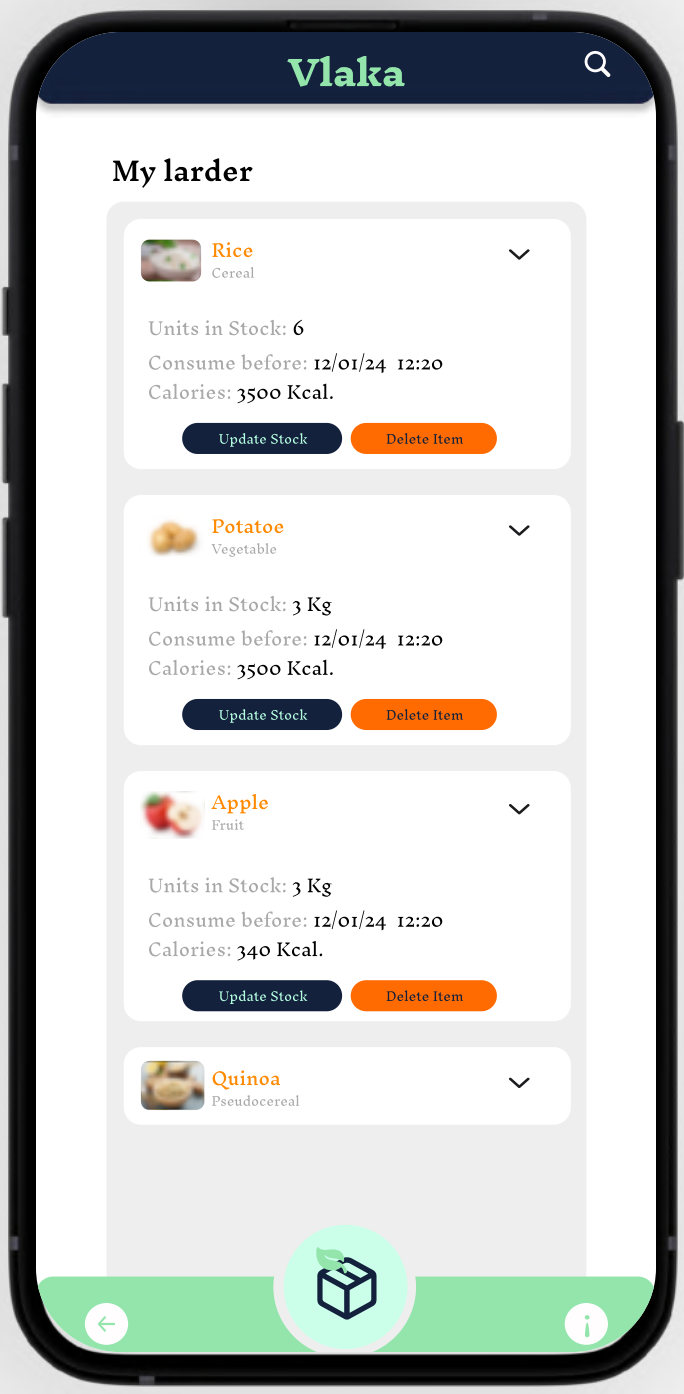
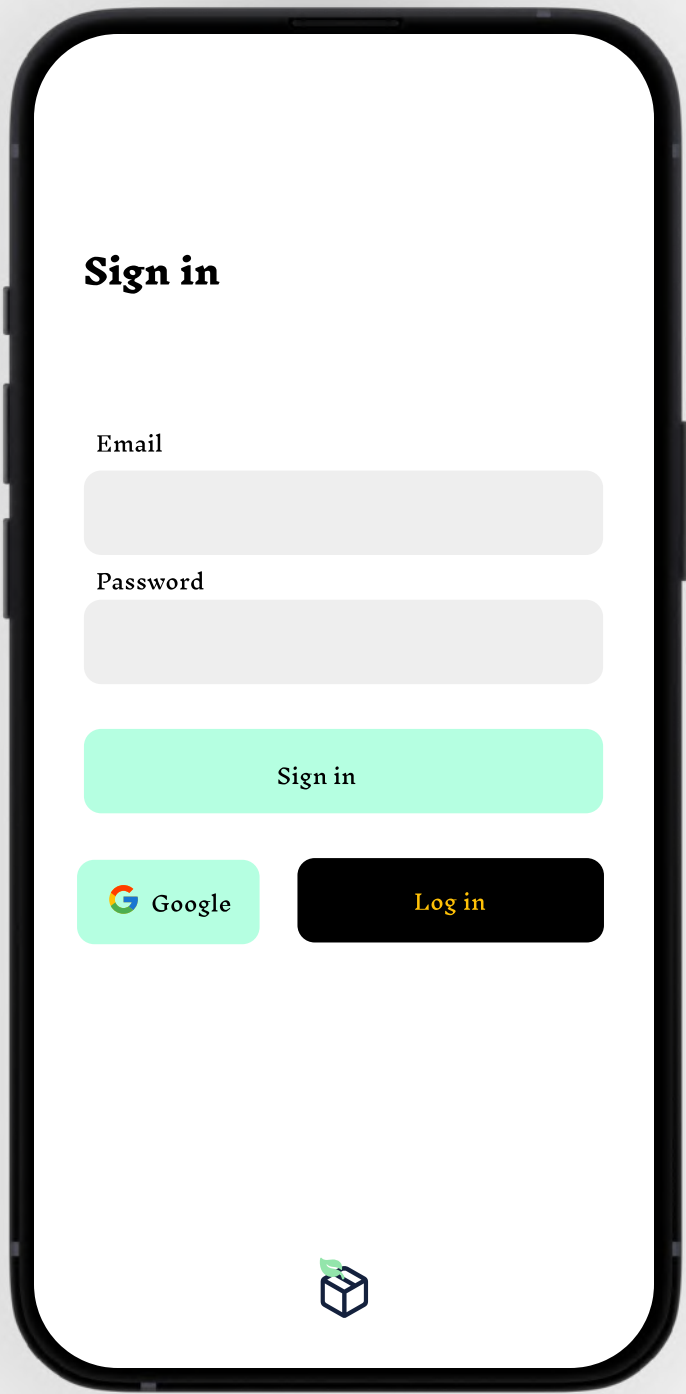
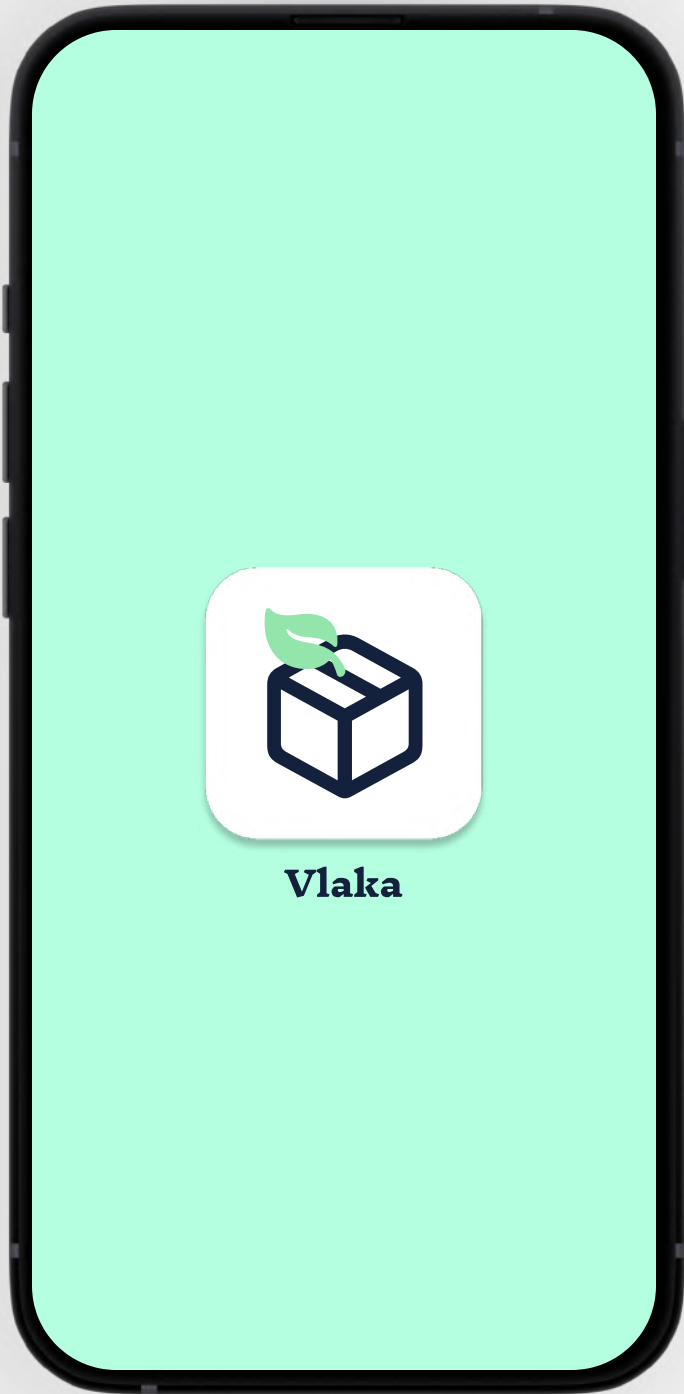
Solución



Despensa de alimentos disponible **24/7**

Vlaka está diseñada para que puedas organizar tu despensa de forma **cómoda**, siempre a mano y **donde quieras**. Con un **diseño moderno** y **fácil uso**, y muy **reusable**

Previsualización Mockups



```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    tools:context=".views.SplashFragment"...>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" ...>

        <androidx.cardview.widget.CardView
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:layout_gravity="center"
            android:backgroundTint="@color/white"
            app:cardCornerRadius="36dp"
            app:cardElevation="20dp">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:src="@drawable/ic_vlaka_v" />

        </androidx.cardview.widget.CardView>

        <TextView
            android:gravity="center"
            android:paddingTop="15dp"
            android:text="Valka"
            android:textColor="@color/azul_oscuro"
            android:textSize="20dp" .../>

    </LinearLayout>

</RelativeLayout>

```

Código

Parte Visual (1/2)

En la parte visual (XML) hemos utilizado contenedores del tipo `<LinearLayout/>`, `<CostrainsLayout/>`, `<RecyclerView/>`, `<FragmentContainerView/>`, `<CordinatorLayout>`, `<RelativeLayout/>` etc..

Los componentes que hemos utilizado son los siguientes:
`<CardView/>` es un elemento visual en forma de tarjetas de información;
`<Button/>`, `<BottomAppBar/>`, muchos `<TextView/>` y `<TextInputLayout>`.


```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    tools:context=".views.SplashFragment"...>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" ...>

        <androidx.cardview.widget.CardView
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:layout_gravity="center"
            android:backgroundTint="@color/white"
            app:cardCornerRadius="36dp"
            app:cardElevation="20dp">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:src="@drawable/ic_vlaka_v" />

        </androidx.cardview.widget.CardView>

        <TextView
            android:gravity="center"
            android:paddingTop="15dp"
            android:text="Valka"
            android:textColor="@color/azul_oscuro"
            android:textSize="20dp" .../>

    </LinearLayout>

</RelativeLayout>

```

Código

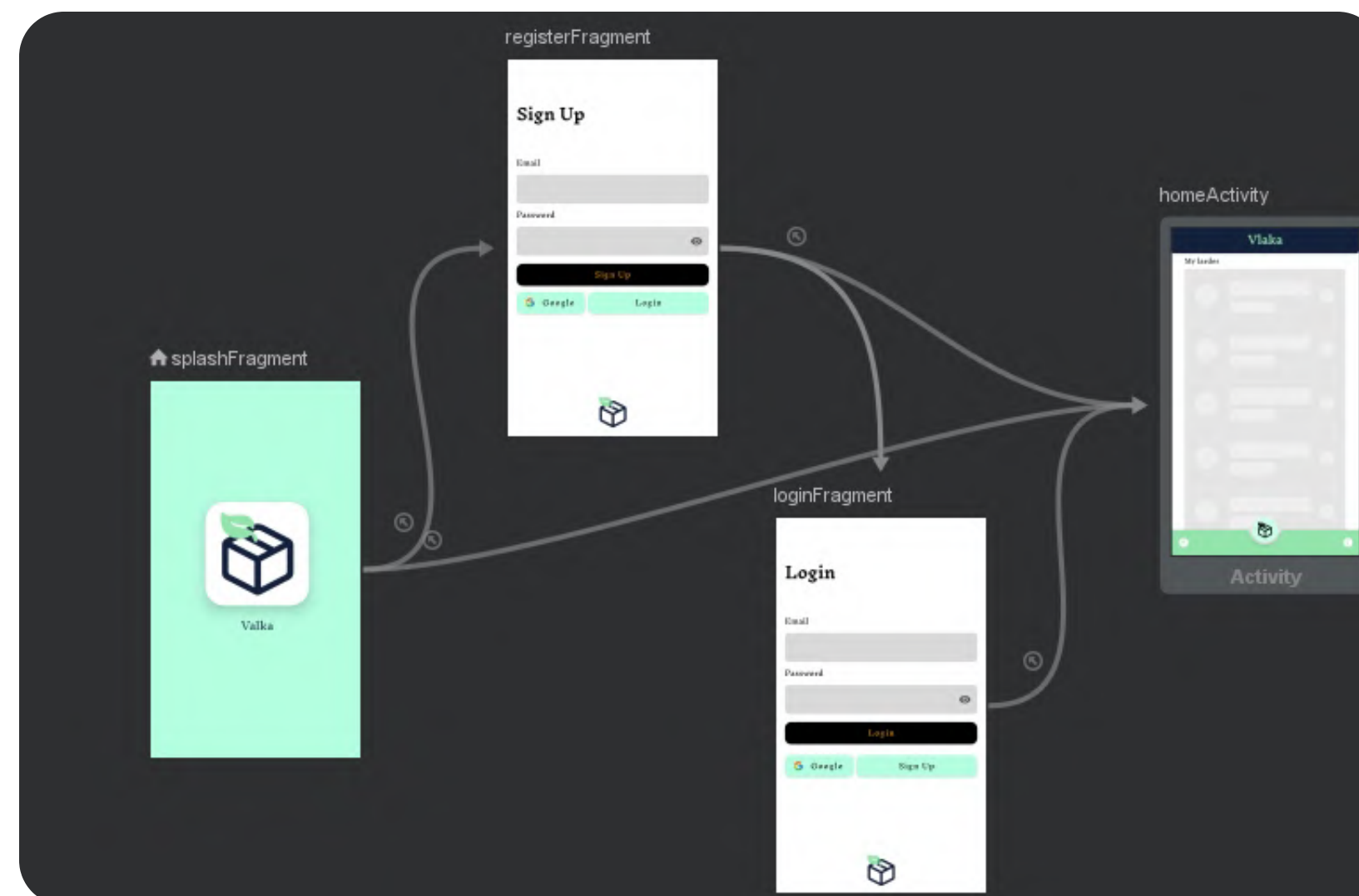
Parte Visual (2/2)

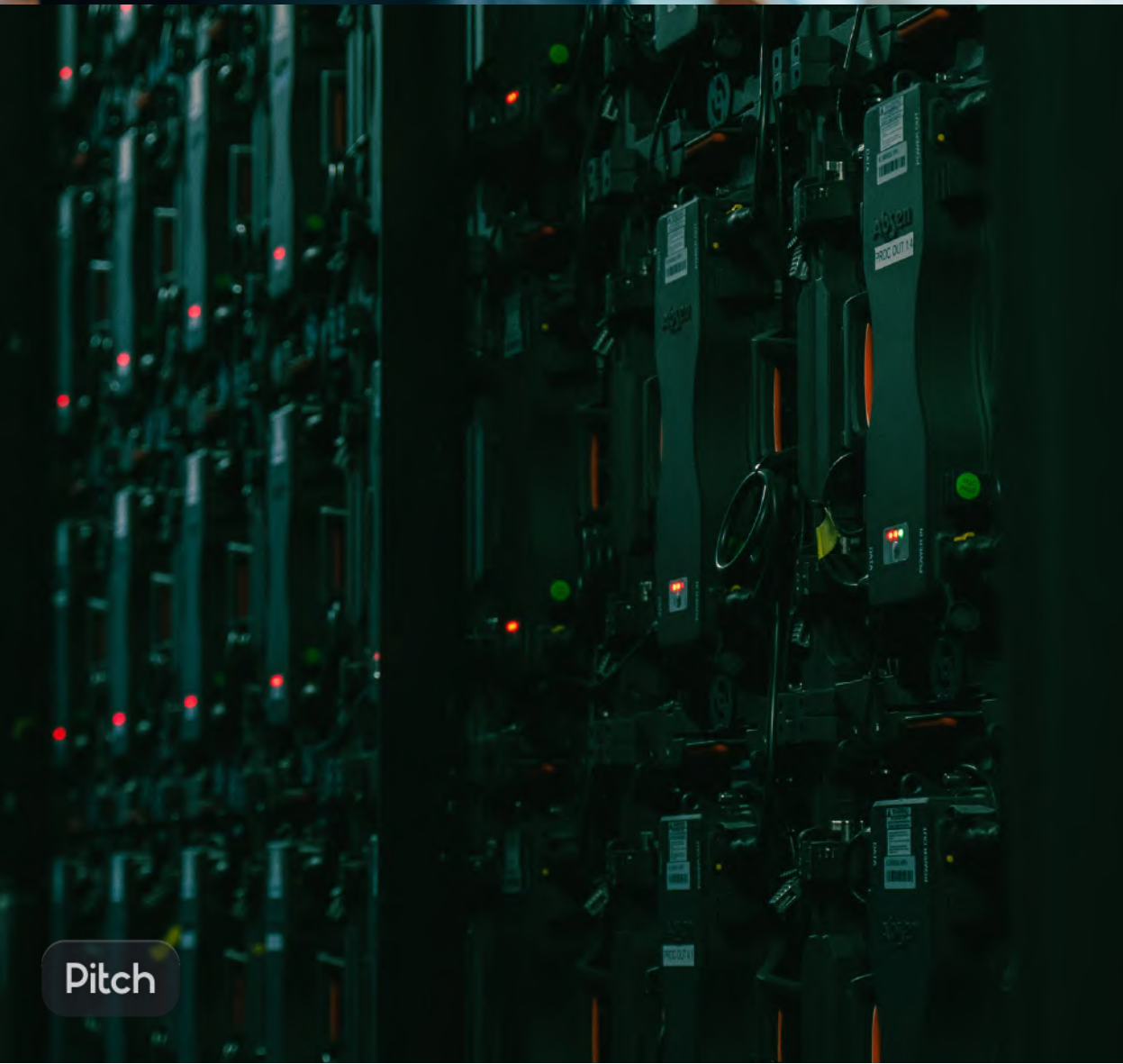
Otros componentes utilizados han sido: `<Chip/>` sirve para crear selectores más visuales, se comporta como un `RadioButton`, es decir que siempre haya un elemento seleccionado así como `<SwitchMaterial/>` y otros componentes como `<DatePicker/>`, `<TimePicker/>` o `<GifImageView/>`

Código

Navegación

Hemos hecho uso del **gráfico de navegación** asociado a un NavHostFragment. En el gráfico de navegación, se especifican todos los destinos a los que el usuario puede navegar en este NavHostFragment y es más visual.





Código

Authentication

Para la autenticación hemos utilizado Firebase. En nuestra aplicación podrás registrarte con un correo electrónico o simplemente autenticarte con Google.

Para realizar el guardado de los productos añadidos desde la app hemos utilizado **Firestore Database**.

También hemos utilizado **Storage** para la librería de fotos.


```

class LoginFragment : Fragment() {

    private lateinit var auth: FirebaseAuth

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)
        auth = FirebaseAuth

        val btnStart = view.findViewById<Button>(R.id.btnIniciarSesion)...
        btnStart.setOnClickListener {
            loginUser(email.editText?.text.toString(), pass.editText?.text.toString())
        }

        btnBack.setOnClickListener {
            findNavController().popBackStack()
        }
    }

    private fun loginUser(email: String, password: String) {
        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(requireActivity()) { task ->
                if (task.isSuccessful) {
                    Toast.makeText(context, "Logueado", Toast.LENGTH_LONG).show()
                    findNavController().navigate(R.id.action_loginFragment_to_homeActivity)
                } else {
                    Toast.makeText(context, "Error en login", Toast.LENGTH_LONG).show()
                }
            }
    }
}

```

Código

Funcionalidad (1/2)

Para hacer las vistas funcionales hemos utilizado:

`<Fragment/>` para la pantalla del Splash, Register y Login.

`<Adapter/>` lo hemos utilizado para la creación de la lista de los productos. El adapter nos ayuda a gestionar el `<RecyclerView/>` `<ViewHolder/>` es el encargado de asignar los valores visuales.

Código

Funcionalidad (2/2)

`<Adapter/>` lo hemos utilizado para la creación de la lista de los productos. El adapter nos ayuda a gestionar el `<RecyclerView/>` `<ViewHolder/>` es el encargado de asignar los valores visuales.

También lo hemos utilizado para la funcionalidad de escoger la fecha y la hora dentro la aplicación.

```
class DatePickerAdapter (val listener: (day: Int, month: Int, year: Int) -> Unit) :  
    DialogFragment(), DatePickerDialog.OnDateSetListener {  
  
    override fun onDateSet(p0: DatePicker?, p1: Int, p2: Int, p3: Int) {  
        listener(p3, p2, p1)  
    }  
  
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {  
        val c = Calendar.getInstance()  
        val year = c.get(Calendar.YEAR)  
        val month = c.get(Calendar.MONTH)  
        val day = c.get(Calendar.DAY_OF_MONTH)  
        return DatePickerDialog(activity as Context, this, year, month, day)  
    }  
}
```

Código

| Class Product |
|-------------------------|
| id (<i>String</i>) |
| image (<i>String</i>) |
| name (<i>String</i>) |
| type (<i>String</i>) |
| stock (<i>Int</i>) |
| time (<i>String</i>) |
| hour (<i>String</i>) |
| calories (<i>Int</i>) |

| Class InfoAboutLarder |
|--|
| numProducts (<i>Int</i>) |
| categories (<i>ArrayList<String></i>) |
| totalCalories (<i>ArrayList<String></i>) |



Problemas con los cuales nos hemos encontrado

- ✓ `<Styles/>` que están ahí pero para no funcionar.
- ✓ `<RecyclerView/>` se ponía encima del `</CoordinatorLayout>` y algunos elementos del RW no se llegaban a ver
- ✓ Al hacer el log out tuvimos problemas para navegar desde un Activity a un Fragment
- ✓ La manipulación y el color de los Chips choice
- ✓ Botón de iniciar sesión que inicia sin cuenta.
- ✓ Autenticación con Google si, pero no hace falta poner correo y contraseña.

Futuras Mejoras



Diseño

Un mejor diseño de interfaz.



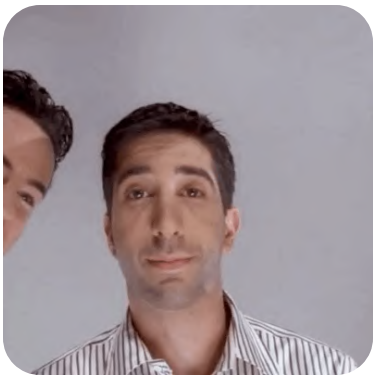
Notificación

Notificación para un producto que se vaya a caducar.



Beneficiencia

Alimentos que sobren o que no quieras y que puedan ser donados



Grupos

Tener un grupo para que todos los integrantes de la casa sepan que hay.



Dinero

Promedio del gasto mensual en alimentos



Migración a nevera

Más y mejor



¿Dónde lo compre?

Así recordarás donde compraste ese producto delicioso



Filtro por Alimentos

Así será mas fácil encontrar lo que estés buscando



Vlaka

¿Dudas y Preguntas?

Thank you!

